ATARI COMPUTER ENTHUSIASTS

3662 Vine Maple Dr. Eugene OR 97405

DECEMBER, 1985

Mike Dunn, Jim Bumpas, Larry Gold, co-editors

JANUARY/1986

HAPPY HOLIDAYS

FROM ACE

News and Reviews

by Mike Dunn, Co-Editor

The last meeting of ACE was exciting in that we had a comparison of the **Amiga** vs the **Atari ST** . A local Amiga dealer was kind enough to come and demonstrate their fine machine. Very impressive, but for more than twice the price of the ST, not that much different. It was eresting to see the famous bouncing ball on both the machines. The main difference seemed to be the multitasking and animation of the Amiga, but I think "Power without the price" won the day. I do not have an ST, but may get one someday. I find the 8-bit Ataris powerful enough for my needs. Remember our Dec. meeting features our annual **swap meet** with proceeds to our local science museum, WISTEC.

The big club news is that we have just received our modem from **U.S. Robotics** , the 300/1200/2400 Courier. They have a very special deal for bulletin boards of clubs such as ours to buy one at a very low price. We are also setting up a 10 Meg Hard Disk, and hope to be operational soon. Larry Gold, Vice Pres and SYSOP will tell more.

We have some new disks for the library, a new line of old and new programs for the new Atari 8-bit owner. Two are now almost ready, each on a double sided disk for only $10. Incidentally, a double-sided disk means that each side of the disk is filled with programs, and you turn the disk over to use the other side. It does not require a special drive; you do get twice as much for your money. These disks, called **The Essential ACE** disks will be offered to new members, but anyone may order them. They include many utilities by your ACE favorites including Stan Ockers, Dale Lutz, Paul Freeman, Bob Floyd (of SPACE), Sydney Brown and from our U.K. friends, Page 6 and the U.K. Atari Club Monitor, and others.

Essential ACE Disk One:

Side 1: The complete DOS 2.5, the patch to make it memory resident in an XL, the Translator file to run old Atari programs on a XL or XE, several different disk and cassette copy programs, including XECOPY (allows you to copy an entire disk with a XE with one pass), and a very nice, simple disk directory and label maker program from our Pres., Dick Barkley.

Side 2: MACH DOS with documentation, a disasmbler, a charactor generator, a PMHelper generator, a sector editor, a renumber program and similiar additional utlities.

Essential ACE Disk Two:

Side 1: A modified DOS 2.0 called 2.6f, that has several very nice features, and various printer utilities. Included is Pictrix, in a compiled version by Paul Freeman, a program that allows you to load and interchange as well a print out pictures from MicroPainter, Atari Touch Tablet, etc. From Bob Floyd comes NiceList in compiled versions for eral printers plus the source code to modify, a very nice program rint out the funny control charactors to your printer, a MX80 graphics dump and a disk label program.

Side 2: Has DOS 2.5, AMODEM 7.1, the newest and best public domain modem program, drivers for Atari and MPP modems, complete documention, 40 and 80 column versions of Stan Ocker's TinyText, and his Label program.

We have also just received a set of 4 ACTION! Utility Disks and 1 ACTION! Game disk from **NovAtari** club of Virginia, a great collection of programs including a Pseudo-Assembler, a Public Domain Run-time generator, a program to analyzes your program to see what run-time modules you need, and many other useful things. The disks are documented well, and until we organize them and put them in our library, for this issue, you can get the 4 Utility disks for only $20 (2 double-sided disks. Highly recommended for intermediate to advanced ACTION! programmers.

We have just received the **Print Shop Graphics Library -3** from Broderbund, which includes 120 new graphics for christmas, myth and fantasy, seasons, animals, international symbols, animals, an eye chart, etc. Recently, my daughter who has never seen **PrintShop** and one of her friends needed to make some posters and I suggested using this program. Without looking at the manual, they were able to produce some great looking posters and have a blast doing it. If you have a graphic printer, you are missing a great deal if you don't get this program for christmas to at least make some christmas cards (get Graphics #3 also). It is fun and so easy that even an adult can do it!

If you are interested in **PrintShop** graphics, several Atari User Groups have gotten together and are putting together disks for $10 each. If you are interested, send me your best efforts, and I'll send you a disk full when it is completed and then trade with the other groups.

One of the good friends of the Atari world, **Gary Furr** the designer of **Atari Writer** and the developer of the printer drivers for this fine word processor is now alone and marketing the various drivers. Because of the low demand, probably mainly because people do not know of them, Gary is getting discouraged and is thinking of stopping his support. If you have the AtariWriter and want it to work the best for your printer, send him only $10, and your printer will really reach it's capabilities. Gary was a great supporter of the user groups in the "old" Atari company, going out of his way many times to help us. He deserves our support. Rembember, only $10 for almost any printer; let him know which one you want and that ACE sent you. (Gary Furr, POB 1330, Mt. View, CA 94042-1330).

In the last issue, the excellent programs from England, Sector and Computer Aided design were from the U.K. Atari Club as the listing indicated, not Page 6. Sorry for the mixup.

CODE BLOCKS

Action! Code Block Generator

(by Jim Patchell; reprint: SBACE, Sept., 1985)

For those who don't know, code blocks are a way of putting machine language programs into your Action! programs. They are also a nice way to add efficient library functions to your Action! programs. In fact, the Action! Run Time Library is noting but a collection of code blocks which perform various functions. A good example of what a code block is is the BYTE FUNC ConCat at the top of the program listing. All the hex numbers surrounded by the [ ] characters are a code block. the Action! Compiler converts these numbers directly into binary code and puts them into the computer. The special way of defining the function allows us to call these code blocks from our Action! programs.

In the past I have made smaller functions which I could call, such as the function BYTE FUNC hi, and these are very easy to enter in by hand. But the big string handling functions were first written and tested with an assembler and debugger. After they worked, I took the assembly listing and entered by hand the object portion of the listing. Tedious City There had to be a better way!

I wrote this program for doing this job. The program requires a standard binary saved object file. It takes that file and starts outputting a series of code blocks until it is done. There are parts of this program which actually do nothing. There is a section of code which tries to figure out how long the code is, but I find that some of the files AMAC generates do not have the proper header, so this information is not used in the part of the program which makes the blocks. If you are using AMAC, you may have some extra stuff tagged onto the end of the code block, so use your Action! editor and your assembly listing to get rid of the extra stuff. Now all you have to do is just define your PROC or FUNC and you will have a new library function you can call.

By the way, only assemble one routine at a time. If you have a whole bunch in one file, you will only get one big bunch of codeblocks. This program cannot separate out seperate procedures.

Also, if you wish to do less typing, BYTE FUNC SubPos and DelSub are not used, but you may want them anyway. Just be careful not to make any mistakes when typing in a code block.

DEC MEETING AND SWAP MEET WED. 11TH DEC. 7:30 PM AMAZON PARK

DISK DRIVES

(by Bill Petry; reprint: RAG BAG, June, 1985)

Like most "intelligent" peripherals (printers and the 850 interface), Atari disk drives contain a 6507 microprocessor (MPU), a 6532 Peripheral Interface Adaptor or PIA, (due to the 6507's lack of direct connections), an Operating System in Read Only Memory (ROM), and alittle Random Access Memor (RAM) to serve sector buffer and scratch pad needs. These process all commands and control data flow to and from the computer.

What sets the disk drive apart from other peripherals is the Floppy Disk Controller (FDC) chip. The FDC controls data to and from the diskette through the Read/Write Erase Head.

The R/W Head is "stepped" from the outer track 0 to the innermost track 39 by the Stepper Motor which is controlled by the MPU through the PIA. The head is moved a step at a time in or out to align it with the currently desired track.

The Drive (or Spindle) motor spins the diskette at a speed of 288 rpm. It has two states: On and Off. It is controlled by the MPU through the PIA.

Initialization on the 810 goes something like this: On power-up the MPU does a JMP (GOTO) to the address contained in the last two bytes of the program in the ROM chip. This starts the program execution and does the following: a) turns on the drive motor; b) positions the head to track 0; c) waits for a command from the computer, or after 7 seconds steps in to track 39 and turns off the drive motor. This command could be either the loading of a boot program or DOS. But the computer must send the command first (the Archiver Chip excluded).

There are five valid commands for the stock 810 drive: Format disk, Read sector, Write sector, Sector status, and Put byte (no verify). The 1050 drive has an additional command for the Medium Density format.

Enhanced drives (Happy and Archiver) and the un-released "E" version of the 810 ROM support other commands. The E ROM permits downloading of user programs. The Archiver Chip supports 16 commands including Set Drive shutdown time, Open chip, Read sector address, Write CRC (Cyclic Redundency Checksum), and Download, as well as the original five.

The U.S. (Ultra Speed) Doubler ROM for the 1050 supports true double density (256 bytes/sector), the original five commands, plus ultra-speed sector skewing, 35/40 tracks per inch spacing and a few others.

The 810 drive has a 1771 FDC and can support only single density format. The 1050 drive contains a 2793 FDC which can support single and medium density (128 bytes/sector and either 18 or 26 sectors per track), and true double density (256 bytes/sector at 18 sectors per track).

Percom drives have a 6809 MPU, 6850 PIA and a 1795 FDC. The single Rana I examined contained a 8031 Control oriented MPU with RAM and I/O, an 8155 HMOS RAM I/O port timer, and a 2797 FDC.

A more detailed account of each part of the drive's electronic and mechanical units is provided in the 810 and 1050 Disk Drive Field Service Manuals. The Atari 810 Disk Peripheral Device Description (15-DEC-80) provides a bit more insight.

The 810 drive contains 4 separate printed circuit (PC) boards: a Power board in the back; an analog board on top of the drive mechanism itself; and a two-parts-in-one side board.

The Power supply board provides for rectified 5 and 12 volts for motors, etc., and motor speed control circuitry. You can adjust the rpm (with the proper program) y turning the screw on the green potentiometer (or the flat white one to the rear left on the older models). The improved power boards provide very stable speed control.

The Analog board (missing on the early models) provides better read/write amplification and cleaner signals.

The side board assembly contains the actual "controller" (like the Apple controller card), with the MPU, ROM, RAM, PIA and FDC. The small board inside the Radio Frequency (RF) shield is the external data separator.

The FDC encodes data on the diskette serially (one bit at a time) in FM (not stereo though!) with a reference 4-microsecond clock pulse with each bit written.

The letter "R" ($52, #82) would be something like the following:
C0 C1 C0 C1 C0 C0 C1 C0

The C's indicate the clock pulses (which actually are recorded like the 1's). The "R" (#82, $52) is 01010010 in binary.

During a diskette read the signal is received from the diskete and the data must then be separated from the clock pulses to make useable data bytes. The 1771 FDC contains an internal data separator, but due to timing tolerances with the 5.25" diskettes on the inner tracks, it is inadequate to readily handle the bit shifting. The result is the familiar "Time Out" and its associated irks. The external separator compensates for the shifting, keeping the current bits properly "windowed" and the flow of data smoother.

The 1050 works basically the same, although there is only one PC board with all the electronic components on it. The medium density is recorded slightly different electronically on the diskette also.

The above is a sketch of the . . . Disk Drive SIG [of the Redwood Atari Group] . . . .

FLEA MARKET

This year's annual pre-Christmas Flea Market to benefit WISTEC will be different than in previous years. First of all, we don't ask you to donate 10% of whatever money you receive for items you sell. This year we ask only a flat donation of $5.00 per person (unless you're only a buyer — buyers get in free).

So if you have some hardware you're no longer using, bring it and see if someone wants it. If you have some original software with original documentation you're no longer using, someone else might want to buy it. Books, magazines and any computer-related item is welcome.

Contact Larry Gold if you want to sell any items and give him the donation for WISTEC. You will receive a receipt from WISTEC for your donation (you don't need to limit it to $5.00 if you choose to give more to WISTEC).

FURTRADER

```
 ROBBED YOUR PACK       HORSES OF SOME
OF YOUR FURS."
7140 GET #1,Y
7150 ? "5"
7200 IF INT(RND(1)*125))5 THEN 7250
7210 M=M-INT(RND(1)*(M/2))
7220 POSITION 0,9:? " SOME ROBBERS HAV
E JUST ROBBED YOUR      PACK HORSES OF
SOME OF YOUR MONEY."
7230 GET #1,Y:? "5"
7250 GOTO 25
8000 ? "YOU DIED DUE TO A LACK OF REST
.":GOTO 8030
8020 ? "BAD LUCK.YOU STARVED TO DEATH.
"
8030 ? :? "YOU HAD ";F;" PACK HORSES O
F FOOD AND WATER LEFT."
8060 ? "YOU COULD HAVE SURVIVED ";D;"
DAYS WITHOUT REST."
8080 ? "YOU COULD HAVE SURVIVED ";W;"
DAYS WITHOUT FOOD/WATER."
8100 ? :? "IN YOUR WILL YOU LEFT :"
8115 M=M+(HF*HFP)+(CF*CFP)+(OF*OFP)+(R
F*RFP)
8130 ? :? "A TOTAL OF &";M;" NET"
8150 ? :? "( ";HF;" HAGGIS FURS )"
8160 ? :? "( ";CF;" COYOTE FURS )"
8170 ? :? "( ";OF;" OCELOT FURS )"
8180 ? :? "( ";RF;" RABBIT FURS )"
8220 GET #1,Y
8230 ? "5TO PLAY AGAIN PRESS ANY KEY."
:GET #1,Y:GOTO 6
9000 POSITION 0,17:? "          WHAT AB
OUT COMMAND (5) ?       ";:GET #1,Y:GOT
O 25
9020 POSITION 0,17:? "  PLEASE ENTER A
 CORRECT COMMAND !!     ";:GET #1,Y:GOT
O 25
9040 POSITION 0,17:? "  YOU ARE NOT AT
 A TRADING POST  !!     ";:GET #1,Y:GOT
O 25
10000 POKE 710,55:SOUND 0,36,36,36:FOR
 J=0 TO 100:NEXT J:POKE 710,148:SOUND
0,0,0,0:TRAP 10000:GOTO 25
```

ORANGES

The Saga of ORANGES

by Dale Lutz

In June, 1984, I had the great idea to do an article comparing several Atari high level languages. What better way to do this, I thought, than writing a simple game in each of the languages, and then pointing out the important differences between each of them. Well, the game I chose to write is entitled 'ORANGES', and it is your very simple 'things are falling so catch them' type of game. I wrote the game first in BASIC, then in ACTION, then in FORTH, and finally in ACE C. Before I start in with the details of each of the languages, I should first give an overview of the game itself.

The player's task in ORANGES is simple. An orange (actually a graphics 5 pixel) is falling and you position your basket below it to catch it and earn points. If you miss 3 oranges, the game ends. A high score tally is kept and displayed on a Graphics 1 text window on the bottom of the screen. So basically all the program does is 1) set up the screen, 2) wait for START to be pressed, and then 3) drop oranges until 3 are missed, and repeat the whole process.

I first wrote the program in BASIC because at the time it was the language I knew the best, and because it is interactive and interpretive, meaning you can make changes to the code on the fly without needing to recompile. All the other languages require a recompiling process that takes time and can be a bother. Recompiling ACTION! programs is the least bother because the compiler is built into the cartridge and you can have the source code in memory at the same time as the compiled version is running. FORTH is quite a bother to recompile because first you have to use the editor to edit a 'SCREEN' (FORTH has no notion of a source file — you must put your code onto blocks on the disk called SCREENs, and then load these screens back in to have them compiled), and then LOAD the screens back in. Of course, as the screens load in, they are compiled and all the procedure names (called VERBs in FORTH lingo) are added to the dictionary (the library of available FORTH verbs). The problem is that if you are loading the program a second or third time, you keep adding more and more identical verbs to the dictionary, all the time getting the message 'ORANGES is not unique' message for each verb that was already there. To recompile C is also a bit of a hassle because you have to first load in some kind of editor to change your source file, then load in the compiler to compile it, and finally load in the linker to link your source file to the routines it uses (like DRAWTO etc.) With the Ramdisk on a 130XE this isn't so bad, and I understand a new version of C that makes use of the XEs extra memory is available too, but with an old 800 it is a very slow process to recompile.

After writing the program in BASIC I translated it to ACTION. Going from BASIC to ACTION was a bit tricky as I changed all the subroutines in BASIC to PROCedures in ACTION!. Another noticeable change was that all the variables I used in BASIC had to be declared at the top of the ACTION program. At first I thought that was a very dumb idea, but as I become more experienced of a programmer I find that not having to declare variables only leads to bad practices and big troubles in debugging. However, it was quite straightforward going from BASIC to ACTION, and the code produced by both is quite similar. In my opinion, the ACTION code is the easiest of all four to read because of its nice WHILE condition DOOD structures, and its familiar BASIC-like commands. ACTION has a nice shorthand for adding a value to a variable that is a bit different. In BASIC you have, say, BASKETS = BASKETS + 3, in ACTION! you only need to write BASKETS = = + 3.

Next I translated the program into FORTH. If you look at the FORTH listing, you will notice two things. The first is that it hardly takes any room at all, the second is that you cannot figure out what it is doing. The constant and variable declarations are straightforward enough, but from then on it gets crazy. FORTH verb definitions always start with a colon, then the verb name then what it is to do, terminated with a semicolon. Understanding FORTH requires you first to grasp the idea of postfix notation. FORTH uses a stack, and all arithmetic you perform is done on this stack. Examples are the best way of illustrating: Suppose you want to add the numbers 3 and 4. In FORTH you would type 3 4 +. This way of expressing things is used by many scientific calculators, and has the advantage that parantheses are not needed. Many people who are used to it swear by it, but I feel that for the most part it is unnatural to us, and consequently hard to read. Anyway, the effect this has on your FORTH code is that everything is backwards to what you would think. I will give one more example: Suppose in BASIC you had this line: IF J = 3 THEN BASKETS = BASKETS-1. In FORTH, this becomes: J @ 3 = IF BASKETS @ 1 - BASKETS ! THEN. The @ gets the value of the variable, and the ! stores a value in the variable.

My biggest complaint about FORTH is the method for storing your code. Because of this screen oriented source storage, inserting a couple of lines in the middle of something is not a trivial job. If FORTH would use normal files for its source, I think one could grow to like it because the code is so compact. But as it stands, it is a nightmare to program in. I translated the ACTION version directly to FORTH, and it got it working quite fast. But if I were just making the program up as I was going, it would have been impossible because I wouldn't have been able to easily add chunks of code I forgot about as I was trying to debug.

This past spring I translated ORANGES into C. Again I went from the ACTION! version to C. In fact, I used the same source file and only modified it in the places where C and ACTION differed. C is not a bad language either — the biggest drawback is the editor/compiler/linker switching all the time, however, this is only a fault of our old 8 bit machines. C will be of the most interest to those of you who are going for the new STs. I will pass on a few things I picked up while doing my translation. First and most important, all C reserved words (like IF, etc.) MUST BE IN LOWER CASE. You can name your functions in upper case if you want, but any C words must be in lower case. ACTION is forgiving about cases, but C is not. Another big difference is that each C statement must be terminated with a semicolon. Note that a statement is not the following: while (Strig(0) = = 1). If a semicolon is after this, the code you want executed while the condition is true will not be executed. Also notable about C is that it makes a difference between its assignment operators, and its test for equality. To assign a value, you do as in BASIC (baskets = 0), or whatever. However, to test for equality, you type baskets = = 0. This is because in C you may assign and test for equality on the same line. Also C has this notion of a code block. A code block is equivalent to the code between the DO and OD in ACTION. Each procedure is considered a code block. Code blocks are normally set off in C by opening and closing braces, but since on our 8 bit machines we don't have these, we must use the $(and $) to set off code blocks. Make a note that it is $(and NOT ($ — I had them switched for while and couldn't figure out why it wouldn't compile.

The final point I want to raise is the running times of the completed programs. The BASIC program runs the most slowly, as one might expect since BASIC is an interpretted language. Even with no delay built in, the BASIC program is so slow that playing it is no challenge. The FORTH and C programs have very similar running speeds. A slight delay was put in both to make them playable. The ACTION! version runs the fastest of all. Even with a large delay loop built in, the program is very challenging and almost too fast.

Those of you so inclined should be able to learn quite a bit while studying the listings of the program in the different languages. I sure learned alot while programming them. If you are really interested, you might want to consider getting FORTH and C for yourself. I beleive that ANTIC sells a documented version of FORTH, and ACE C is available from the ACE library for a minimal charge. (Public domain versions of ACE-C and fig-FORTH from S.P.A.C.E, with no documentation for $10 each from ACE, or the super XE-C from Ralph Walden for $35). The 130XE C is available from Ralph Walden, and is recommended for those with an XE. A great book for learning FORTH is Leo Brodie's 'Starting FORTH'. I can't say enough about how good this book is — if only all computer manuals were this good. A book that can be tough sledding but will tell you all about the C language syntax and features is ''The C Programming Language'' by Kernighan and Richie. It is the final word on any question one may have about C.

In closing, I want to encourage those of you who are tired of BASIC, and are looking for a new challenge, to get one of these languages, and learn it. Even if you don't have any practical use for it, it is alot of fun to learn a new language, and the experience you will gain will help you alot in the future. This summer I had to learn FORTRAN where I worked, and my experience with these languages really helped me.

Ralph Waldon's **130XE-C** is available with extensive documentation for $35 from him a 1821 Jefferson, Eugene, OR 97402. Specify single or double density. According to some C experts I know, it is better than any available for the IBM-PC, etc. [-ed.]

CALENDAR

DALE LUTZ

```
; Calendar Generator/Date Handler

; By Dale Lutz
; July 31, 1985

; To custimize the sizes of the calend
ars,
; you should change the variables
; 'depth' and 'width' in the PROCedure

; Calendar.  Additionally, you may wis
h
; to modify the printer setup strings
; to use larger printing or to work on

; a non-Epson printer.

; I like to use the large print, large

; style calendar for keeping track of
; appointments, and the small print,
; small style calendars for in the
; pocket reference.

; The BYTE FUNCTION GetChoice is a
; 'Print Shop' type menu easily usable

; in other programs.  It can be moved
; easily to other programs.  It is
; documented within its body below.

INCLUDE "D8:SYS.ACT"

DEFINE year="1985",
       month="01",
       day="01",
       vbar="156",  ; '| on the screen

       hbar="157"   ; '- on the screen

      ;vbar="'|", ;used when testing
      ;hbar="'-"  ; "    "      "


BYTE ARRAY daym=[00 31  0 31 30 31 30
                 31 31 30 31 30 31],

; the following two byte arrays are th
e
; printer setup.  The first byte is th
e
; length of the setup string.  The nex
t
; two reset the printer, then the pape
r
; out sensor is disabled (27 '8),
; unidirectional printing started (27
 'U 1),
; for the small, superscript mode is
;    started (27 '5 0),
; compressed mode entered (15),
; and the line spacing set to 7 or
; 4 72nds of an inch (27 65 [7 or 4]).


     large=[11 27 'e 27 '8 27 'U 1 15
27 65 7 ],
     small=[14 27 'e 27 '8 27 'U 1 27
'5 0 15 27 65 4 ],

     sizem=[0 7 8 5 5 3 4 4 6 9 7 8 8]
,
     route=[2 'P ';]

CARD ARRAY namem(12),named(6),menu(10)

CARD yy
BYTE mm,dd,doomsday,weekday,dayflag

PROC FillStr(BYTE ARRAY str)

BYTE temp

  IF str(0)=1 THEN
    temp=str(1)
    str(2)=temp
    str(1)='0
    str(0)=2
  FI

RETURN

PROC FancyGet(BYTE ARRAY date BYTE x,y
,len)

;This procedure will allow fancy input

;of a date.
;It intercepts all input and
;will take off as soon as either
;the box is full or RETURN is pressed.

;a default date should be in 'date'
;when this routine is called.
;the x,y variables tell the position
;on the screen for the input to occur



BYTE numlen,a,key

Position(x,y)
FOR a=1 to date(0)
  DO
    Put(date(a))
  OD
Close(3)
Open(3,"K:",4,0)
numlen=1
key=0
Position(x+numlen-1,y)
Put(30)
Put(31)
WHILE key()155 AND numlen(len+1
  DO
  key=GetD(3)
  IF key)='0 AND key(='9 THEN
    date(numlen)=key
    Put(key)
    numlen==+1
  ELSEIF (key=126 OR key=30 OR key='+)
 AND numlen)1 THEN
    numlen==-1
    Put(30)
  ELSEIF key=31 OR key='* THEN
    numlen==+1
    Put(31)
  FI
  OD
RETURN

PROC DateGet()

BYTE ARRAY yys(5),mms(3),dds(3)

  StrC(yy,yys)
  StrB(mm,mms)
  FillStr(mms)
  StrB(dd,dds)
  FillStr(dds)
  Position(14,12)
  Print("YYYY  MM")
  IF dayflag=1 THEN
    Print("  DD")
  FI
  FancyGet(yys,14,13,4)
  FancyGet(mms,20,13,2)
  IF dayflag=1 THEN
    FancyGet(dds,24,13,2)
  FI
  yy=ValC(yys)
  mm=ValB(mms)
  IF dayflag=1 THEN
    dd=ValB(dds)
  ELSE dd=1
  FI
  PutE()
RETURN

BYTE FUNC DoomFind()
```

CALENDAR

```
;This function returns the day of the
;week which is the 'doomsday' for a
;particular year.  The convention
;followed is that 0 means Sunday,
;1 is Monday, and so on..

BYTE theday,twelves,offset,base,leftov
er,
     leaps,remainder,doomsday

  base=3 ;year 1900 Wednesday is dooms
day
  leftover=yy-1900
  twelves=leftover/12
  remainder=leftover MOD 12
  leaps=remainder/4
  offset=leaps+remainder+twelves
  offset=offset MOD 7
  doomsday=(offset+base) MOD 7 ;the do
omsday is found

RETURN (doomsday)

BYTE FUNC FindDay(BYTE doomsday)

BYTE doomdate,final,fake

INT offset,theday

  IF (yy MOD 4) = 0 THEN
    daym(2)=29
  ELSE
    daym(2)=28
  FI
  IF mm=1 THEN
    doomdate = 31 ;January, normal yea
r
    IF (yy MOD 4) = 0 THEN
      doomdate=25 ;January, leap year
    FI
  ELSEIF mm=2 THEN
    doomdate=daym(2) ;Februry
  ELSEIF (mm MOD 2) = 0 THEN
    doomdate=mm ;even months
  ELSEIF daym(mm)=31 THEN
    doomdate=mm+4 ;odd long months
  ELSE
    doomdate=mm-4 ;short odd months
  FI
  fake=dd+42 ;required to avoid negati
ves
  offset=(fake-doomdate) MOD 7 ;find d
ifference
            ;from doomdate to real day

  theday=(doomsday+offset) MOD 7
  IF theday<0 THEN
    theday==+7
  FI
  final=theday
RETURN (final)

PROC TellDay()
BYTE ch
  DO
  Put(125)
  Position(3,8)
  PutE()
  PrintE("            ENTER THE DATE")
  PrintE("            ==============")
  DateGet()
  doomsday=DoomFind()
  weekday=FindDay(doomsday)
  PutE()
  PutE()
  PutE()
  Print("The day of the week is==>")
  PrintE(named(weekday))
  Close(3)
  Open(3,"K:",4,0)
  PutE()
  PrintE("      Press space for anothe
r,")
  Print("        RETURN to exit -->")
  ch=GetD(3)
  UNTIL ch=155
  OD
RETURN

BYTE FUNC GetChoice(BYTE number,indent
)

; Global CARD ARRAY menu must hold
; the addresses of the options.  Then
; this function is called, allowing
; the user to use the arrow keys or
; the numbers to choose from the menu.

; BYTE number holds the number of
; options in the menu, while 'indent'
; tells the program how far to space
; over before printing the options.
; The users choice is then returned.
; Note that this routine can be reused

; several times in a single program
; by simply changing the parameters
; before calling it.

BYTE ARRAY temp(30)
BYTE a,b,key,choice,oldchoice
CARD spc
spc="                                "
Poke(spc,indent)
Position(3,10)
PutE()
FOR a=1 TO number
  DO
  Print(spc)
  PrintE(menu(a))
  OD
Close(3)
Open(3,"K:",4,0)
key=0
oldchoice=number
choice=1
WHILE key<>155 DO
  IF key=28 OR key='- THEN
    choice==-1
  ELSEIF key=29 OR key='= THEN
    choice==+1
  ELSEIF key<='9 AND key>'0 THEN
    choice=key-48
  FI
  IF choice=0 THEN choice=number
  ELSEIF choice>number THEN choice=1

  FI
  IF oldchoice<>choice THEN
    Position(3,9+oldchoice)
    PutE()
    Print(spc)
    PrintE(menu(oldchoice))
    Position(3,9+choice)
    PutE()
    Print(spc)
    SCopy(temp,menu(choice))
    FOR b=1 TO temp(0)
    DO
      Put(temp(b)+128)
    OD
    oldchoice=choice
  FI
  key=GetD(3)
 OD
RETURN (choice)

PROC Calendar()

BYTE ch,doomsday,weekday,a,b,width,
   wholewidth,spaces,maxday,block,
   depth,nummonth,holdmm,numprinted,
   maxprint,op

CARD setup

BYTE ARRAY sday(5),
     max=[0 4 2 10 6]
```

CALENDAR

```
  Put(125)
  PutE()
  PutE()
  PutE()
  PrintE("          Calendar Generatio
n")
  PutE()
  PrintE("               Subsystem")
  menu(1)="1. Large Style, Small Print
"
  menu(2)="2. Large Style, Large Print
"
  menu(3)="3. Small Style, Small Print
"
  menu(4)="4. Small Style, Large Print
"
  ch=GetChoice(4,7)
  IF ch=1 OR ch=2 THEN
    width=10
    depth=4
  ELSE
    width=2
    depth=0
  FI
  wholewidth=7*(width+1)+1
  maxprint=max(ch)
  IF ch=1 OR ch=3 THEN
    setup=small
  ELSE
    setup=large
  FI
  Put(125)
  PutE()
  PutE()
  PutE()
  PrintE("          Calendar Generatio
n")
  PutE()
  PrintE("               Subsystem")
  PutE()
  PrintE("        Enter the Starting D
ate")
  PrintE("        ====================
===")
  DateGet()
  PutE()
  PutE()
  Print("      How Many Months==)01")
  Put(30)
  Put(30)
  nummonth=InputB()
  nummonth==-1
  IF nummonth+mm)12 THEN
    nummonth=12-mm
  FI
  Close(2)
  Open(2,route,8,0)
  PrintD(2,setup)
  numprinted=0
  op=ch
  holdmm=mm
  FOR mm=holdmm TO holdmm+nummonth DO
    doomsday=DoomFind()
    weekday=FindDay(doomsday)
    FOR a=1 TO (wholewidth-(sizem(mm)+
6))/2
    DO
      PutD(2,32)
    OD
    PrintD(2,namem(mm))
    PrintD(2,", ")
    PrintCDE(2,yy)
    PutDE(2)
    IF op=1 OR op=2 THEN
      FOR a=0 TO 6 DO
        spaces=(width-2)/2
        FOR b=1 TO spaces DO
          PutD(2,32)
        OD
        SCopyS(sday,named(a),1,3)
        PrintD(2,sday)
        FOR b=1 TO width-2-spaces DO
          PutD(2,32)
        OD
      OD
      PutDE(2)
    FI
    FOR ch=1 TO (width+1)*7+1 DO
      PutD(2,hbar)
    OD
    PutDE(2)
    block=1
    maxday=0
    WHILE maxday(daym(mm)+weekday DO
      PutD(2,vbar)
      FOR ch=1 TO 7 DO
        IF block)weekday  AND block(=d
aym(mm)+weekday THEN
          StrB(block-weekday,sday)
          PrintD(2,sday)
          FOR b=1 TO width-sday(0) DO
            PutD(2,32)
          OD
        ELSE
          FOR b=1 TO width DO
            PutD(2,32)
          OD
        FI
        PutD(2,vbar)
        block==+1
      OD
      PutDE(2)
      FOR ch=1 TO depth DO
        FOR a=1 TO 7 DO
          PutD(2,vbar)
          FOR b=1 TO width DO
            PutD(2,32)
          OD
        OD
        PutD(2,vbar)
        PutDE(2)
      OD
      FOR ch=1 TO (width+1)*7+1 DO
        PutD(2,hbar)
      OD
      PutDE(2)
      maxday=maxday+7
    OD
    PutDE(2)
    PutDE(2)
    PutDE(2)
    numprinted==+1
    IF numprinted=maxprint THEN
      PutD(2,12)
      numprinted=0
    FI
  OD
  mm=holdmm
RETURN

PROC Main()

BYTE ch

  namem(1)="January"
  namem(2)="February"
  namem(3)="March"
  namem(4)="April"
  namem(5)="May"
  namem(6)="June"
  namem(7)="July"
  namem(8)="August"
  namem(9)="September"
  namem(10)="October"
  namem(11)="November"
  namem(12)="December"

  named(0)="Sunday"
  named(1)="Monday"
  named(2)="Tuesday"
  named(3)="Wednesday"
  named(4)="Thursday"
  named(5)="Friday"
  named(6)="Saturday"

  yy=year
  mm=month
  dd=day
  Poke(710,0)
  DO
    Put(125)
```

ORANGES BY DALE LUTZ

```
0 REM ORANGES IN BASIC
1 REM BY DALE LUTZ 6/6/84
10 BALLSTART=210:TONE=300:WAIT=430
20 LEFTONE=130:RIGHTONE=170
30 SCREENSETUP=320:STARTGAME=360
40 LEFT=11:RIGHT=7:DOWN=39:SCREENWIDTH
=77:CATCHCHECK=230:GAMEOVER=410
60 GOSUB SCREENSETUP:GOSUB BALLSTART
70 ST=STICK(0)
75 IF PEEK(764)<>255 THEN GOSUB WAIT
80 IF ST=LEFT THEN GOSUB LEFTONE
90 IF ST=RIGHT THEN GOSUB RIGHTONE
100 COLOR 0:PLOT BALX,BALY:BALY=BALY+0
.5:COLOR 1:PLOT BALX,BALY
110 IF BALY=DOWN THEN GOTO CATCHCHECK
120 GOTO 70
130 REM LEFTONE
140 COLOR 0:PLOT PLATE+1,DOWN:PLATE=PL
ATE-1
150 IF PLATE<2 THEN PLATE=78:PLOT 1,DO
WN:DRAWTO 3,DOWN:COLOR 2:PLOT 76,DOWN:
DRAWTO 79,DOWN:RETURN
160 COLOR 2:PLOT PLATE-1,DOWN:RETURN
170 REM RIGHTONE
180 COLOR 0:PLOT PLATE-1,DOWN:PLATE=PL
ATE+1
190 IF PLATE>78 THEN PLATE=2:PLOT 75,D
OWN:DRAWTO 79,DOWN:COLOR 2:PLOT 1,DOWN
:DRAWTO 3,DOWN:RETURN
200 COLOR 2:PLOT PLATE+1,DOWN:RETURN
210 REM BALLSTART
220 BALX=INT(RND(0)*SCREENWIDTH)+2:BAL
Y=1:RETURN
230 REM CATCHCHECK
240 COLOR 0
250 IF ABS(BALX-PLATE)<2 THEN CATCHES=
CATCHES+1:COLOR 2:GOSUB TONE:GOTO 270
260 BASKETS=BASKETS-1:SOUND 1,100,8,10
:FOR A=1 TO 50:NEXT A:SOUND 1,0,0,0:IF
 BASKETS=0 THEN GOTO GAMEOVER
270 PLOT BALX,DOWN:POKE 656,0:POKE 657
,10:? CATCHES
280 POKE 656,1:POKE 657,10:? BASKETS
290 GOSUB BALLSTART:GOTO 70
300 REM TONE
310 SOUND 1,81,10,10:FOR A=1 TO 30:NEX
T A:SOUND 1,0,0,0:RETURN
320 REM SCREENSETUP
330 GRAPHICS 5:A=PEEK(560)+256*PEEK(56
1):POKE 752,1
340 IF PEEK(A)<>66 THEN A=A+1:GOTO 340

350 POKE A,70:POKE A+3,6:POKE A+4,6:PO
KE A+5,6:GOSUB STARTGAME:RETURN
360 REM STARTGAME
370 BASKETS=3:CATCHES=0:? CHR$(125):PO
KE 656,1:POKE 657,25:? "PRESS fire":PO
KE 656,0:POKE 657,26:? "oranges"
375 SOUND 0,250,10,10:SOUND 1,252,10,1
0
380 IF STRIG(0)=1 THEN POKE 711,(PEEK(
711)+1)*(PEEK(711)<255):GOTO 380
385 SOUND 0,0,0,0:SOUND 1,0,0,0:POKE 7
11,70
390 PLATE=39:? CHR$(125):POKE 656,0:PO
KE 657,2:? "CATCHES 0":POKE 656,1:POKE
 657,2:? "baskets 3"
400 POKE 657,30:POKE 656,0:? "hi:0 ";H
I:RETURN
410 REM GAMEOVER
412 POKE 656,1:POKE 657,10:? BASKETS
415 FOR A=100 TO 200 STEP 10:SOUND 1,A
,10,10:FOR B=1 TO 50:NEXT B:NEXT A
417 SOUND 1,250,12,10:FOR A=1 TO 250:N
EXT A:SOUND 0,0,0,0
420 IF HI<CATCHES THEN HI=CATCHES
425 ? #6;CHR$(125):GOSUB STARTGAME:GOS
UB BALLSTART:GOTO 70
430 REM WAIT
440 FOR A=1 TO 100:POKE 764,255
450 IF PEEK(764)=255 THEN 450
460 POKE 764,255:RETURN
```

A@

```
/* ORANGES IN C    */
/* BY DALE LUTZ 850718 */

#define key 764
#define down 39
#define screen 77
#define left 11
#define right 7

char a,plate,balx,baly,st;
int b,catches,baskets,hi;

Delay()
${
int i;
  for(b=1;b<=3;++b)
    i=i+1;
$}

Wait()
${
int i;
  for(a=1;a<=100;++a)
    Delay();
  poke(key,255);
  while (peek(key)==255)
    i=i+1;
  poke(key,255);
$}

Tone()
${
  sound(1,81,10,10);
  for(a=1;a<=30;++a)
    Delay();
  sound(1,0,0,0);
$}

Leftone()
${
  color(0);
  plot(plate+1,down);
  --plate;
  if (plate<2)
  ${
    plate=78;
    plot(1,down);
    drawto(3,down);
    color(2);
    plot(76,down);
    drawto(79,down);
  $}
  else
  ${
    color(2);
    plot(plate-1,down);
  $}
$}

Rightone()
${
  color(0);
  plot(plate-1,down);
  ++plate;
  if (plate>78)
  ${
    plate=2;
    plot(75,down);
    drawto(79,down);
    color(2);
    plot(1,down);
    drawto(3,down);
  $}
  else
  ${
    color(2);
    plot(plate+1,down);
  $}
$}

Ballstart()
${
  balx=rnd(screen)+2;
  baly=1;
```

```
$)

Catchcheck()
$(
  color(0);
  if ((abs(balx-plate))<2)
  $(
    ++catches;
    color(2);
    Tone();
  $)
  else
  $(
    --baskets;
    sound(1,100,8,10);
    for(a=1;a<=50;++a)
      Delay();
    sound(1,0,0,0);
  $)
  plot(balx,down);
  poke(656,0);
  poke(657,10);
  printf("%d",catches);
  poke(656,1);
  poke(657,10);
  printf("%d",baskets);
  Ballstart();
$)

Startgame()
$(
  baskets=3;
  catches=0;
  color(0);
  plot(1,down);
  drawto(79,down);
  printf("\f");
  poke(656,1);
  poke(657,25);
  printf("PRESS fire");
  poke(656,0);
  poke(657,26);
  printf("oranges");
  sound(0,250,10,10);
  sound(1,252,10,10);
  while (strig(0)==1)
  $(
    Delay();
    poke(711,peek(711)+1);
  $)
  sound(0,0,0,0);
  sound(1,0,0,0);
  poke(711,70);
  plate=39;
  printf("\f");
  poke(656,0);
  poke(657,2);
  printf("CATCHES 0");
  poke(656,1);
  poke(657,2);
  printf("baskets 3");
  poke(656,0);
  poke(657,30);
  printf("high ");
  printf("%d",hi);
$)

Screensetup()
$(
  open(6,28,5,"S:");
  b=dpeek(560);
  poke(752,1);
  while (peek(b)!=66)
    ++b;
  poke(b,70);
  poke(b+3,6);
  poke(b+4,6);
  poke(b+5,6);
  Startgame();
$)

Main()
$(
  hi=0;
  while (5>2)
  $(
    Screensetup();
    Ballstart();
    while (baskets>0)
    $(
    Delay();
    if (peek(key)!=255)
      Wait();
    Delay();
    st=stick(0);
    if (st==left)
      Leftone();
    else
      if (st==right)
        Rightone();
    color(0);
    plot(balx,baly);
    ++a;
    if (a>1)
    $(
      ++baly;
      a=0;
    $)
    color(1);
    plot(balx,baly);
    if (baly==down)
      Catchcheck();
    $) /*end of baskets0 */
```

SWAP MEET

```
/* This gets executed when the game */

  /* is over (equivalent to the */
  /* GAMEOVER routine in BASIC) */

  for(a=100;a<=200;a=a+10)
    $(
    sound(1,a,10,10);
    for(st=1;st<=40;++st)
      Delay();
    $)
    sound(1,250,12,10);
  for(a=1;a<=254;++a)
    Delay();
  sound(1,0,0,0);
  if (hi<catches)
    hi=catches;
  $) /* infinite loop */
$)



; ORANGES IN ACTION
; BY DALE LUTZ 18/6/84

; GLOBAL VARIABLES, ETC.

BYTE a,plate,balx,baly,key=764,st
CARD b,catches,baskets,hi
DEFINE down="39",
  screenwidth="77",
  left="11",
  right="7"

PROC Delay()
  FOR b=1 TO 300
    DO
    OD
RETURN

PROC Wait()
  FOR a=1 TO 100
    DO
    Delay()
    OD
  key=255
  WHILE key=255
    DO
    OD
  key=255
RETURN

PROC Tone()
  Sound(1,81,10,10)
  FOR a=1 TO 30
```

AMAZON PARK

ORANGES

```
    DO
    Delay()
    OD
  Sound(1,0,0,0)
RETURN

PROC Leftone()
  Color=0
  Plot(plate+1,down)
  plate==-1
  IF plate<2 THEN
    plate=78
    Plot(1,down)
    DrawTo(3,down)
    Color=2
    Plot(76,down)
    DrawTo(79,down)
  ELSE Color=2
    Plot(plate-1,down)
  FI
RETURN

PROC Rightone()
  Color=0
  Plot(plate-1,down)
  plate==+1
  IF plate>78 THEN
    plate=2
    Plot(75,down)
    DrawTo(79,down)
    Color=2
    Plot(1,down)
    DrawTo(3,down)
  ELSE Color=2
    Plot(plate+1,down)
  FI
RETURN

PROC Ballstart()
  balx=Rand(screenwidth)+2
  baly=1
RETURN

PROC Catchcheck()
  Color=0
  IF balx-plate<2 OR plate-balx<2 THEN

    catches==+1
    Color=2
    Tone()
  ELSE baskets==-1
    Sound(1,100,8,10)
    FOR a=1 TO 50
      DO
      Delay()
      OD
    Sound(1,0,0,0)
  FI
  Plot(balx,down)
  Poke(656,0)
  Poke(657,10)
  PrintC(catches)
  Poke(656,1)
  Poke(657,10)
  PrintC(baskets)
  Ballstart()
RETURN

PROC Startgame()
  BYTE flasher=711
  baskets=3
  catches=0
  Put(125)
  Poke(656,1)
  Poke(657,25)
  Print("PRESS fire")
  Poke(656,0)
  Poke(657,26)
  Print("oranges")
  Sound(0,250,10,10)
  Sound(1,252,10,10)
  WHILE Strig(0)=1
    DO
    Delay()
    flasher==+1
    Delay()
    OD
  Sound(0,0,0,0)
  Sound(1,0,0,0)
  flasher=70
  plate=39
  Put(125)
  Poke(656,0)
  Poke(657,2)
  Print("CATCHES 0")
  Poke(656,1)
  Poke(657,2)
  Print("baskets 3")
  Poke(656,0)
  Poke(657,30)
  Print("high ")
  PrintC(hi)
RETURN

PROC Screensetup()
  CARD dlist=560

  Graphics(5)
  b=dlist
  Poke(752,1)
  WHILE Peek(b)<>66
    DO
    b==+1
    OD
  Poke(b,70)
  Poke(b+3,6)
  Poke(b+4,6)
  Poke(b+5,6)
  Startgame()
RETURN

PROC main()
  hi=0
  DO
  Screensetup()
  Ballstart()
  WHILE baskets>0
    DO
    Delay()
    IF key<>255 THEN Wait() FI
    Delay()
    st=Stick(0)
    IF st=left THEN Leftone()
    ELSEIF st=right THEN Rightone()
    FI
    Color=0
    Plot(balx,baly)
    a==+1
    IF a>1 THEN baly==+1 a=0 FI
    Color=1
    Plot(balx,baly)
    IF baly=down THEN Catchcheck() FI
    OD

  ; This gets executed when the game
  ; is over (equivalent to the
  ; GAMEOVER routine in BASIC)

  FOR a=100 TO 200 STEP 10
    DO
    Sound(1,a,10,10)
    FOR st=1 TO 40
      DO
      Delay()
      OD
    OD
    Sound(1,250,12,10)
    FOR a=1 TO 254
      DO
      Delay()
      OD
  Sound(1,0,0,0)
  IF hi<catches THEN hi=catches FI
  OD ; infinite loop
RETURN
```

SWAP MEET AMAZON PARK

FORTH

```
SCR #  20

 00 ( scr# 21   ORANGES      2/? )
 01
 02 : RIGHTONE 0 COLOR plate @ DUP 1
 03 - down PLOT 1 + DUP plate ! 78 >
 04   IF 2 plate ! 75 down PLOT 79
 05 down DRAWTO 2 COLOR 1 down PLOT
 06 3 down DRAWTO ELSE 2 COLOR
 07 plate @ 1 + down PLOT THEN ;
 08
 09 : BALLSTART BEGIN 53770 C@ 2 +
 0A DUP balx ! screenwidth < UNTIL 1
 0B  baly ! ;
 0C
 0D : CATCHCHECK 0 COLOR balx @
 0E plate @ - ABS 2 < IF catches @ 1
 0F + catches ! 2 COLOR TONE ELSE
 10 baskets @ 1 - baskets ! 1 100 8
 11 10 SOUND 50 1 DO LOOP 1 0 0 0
 12 SOUND THEN balx @ down PLOT 0 65
 13 6 C! 10 657 C! catches @ . 1 656
 14  C! 10 657 C! baskets @ .
 15 BALLSTART ;
 16
 17 711 CONSTANT flasher
 18 : STARTGAME 3 baskets ! 0
 19 catches ! 125 EMIT 1 656 ! 25
 1A 657 ! ." PRESS fire " 0 656 !
 1B 26 657 ! ." oranges " 0 250 10
 1C 10 SOUND 1 252 10 10 SOUND BEGIN
 1D  DELAY flasher C@ 1 + flasher C!
 1E  0 STRIG NOT UNTIL  0 0 0 0
 1F SOUND 1 0 0 0 SOUND  -->
```

```
A
 SCR #  21
 00 ( scr# 20  ORANGES      1/? )
 01 DX ( set to decimal mode )
 02 ( define constants )
 03 11 CONSTANT left
 04  7 CONSTANT right
 05 39 CONSTANT down
 06 77 CONSTANT screenwidth
 07 ( initialize variables )
 08 0 VARIABLE a     0 VARIABLE balx
 09 0 VARIABLE st    0 VARIABLE baly
 0A 0 VARIABLE b     0 VARIABLE hi
 0B 0 VARIABLE plate
 0C 0 VARIABLE catches
 0D 0 VARIABLE baskets
 0E
 0F : DELAY 50 1 DO LOOP ;
 10
 11 : WAIT 255 764 C! 100 1 DO LOOP
 12 BEGIN 764 C@ 255 = NOT UNTIL
 13 255 764 C! ;
 14
 15 : TONE 1 81 10 10 SOUND 30 1 DO
 16 DELAY LOOP 1 0 0 0 SOUND ;
 17
 18 : LEFTONE 0 COLOR plate @ DUP 1
 19 + down PLOT 1 - DUP plate ! 2 <
 1A  IF 78 plate ! 1 down PLOT 3
 1B down DRAWTO 2 COLOR 76 down PLOT
 1C  79 down DRAWTO ELSE 2 COLOR
 1D plate @ 1 - down PLOT THEN ;
 1E
 1F -->
```

```
SCR #  22
00 ( scr# 22  ORANGES      3/? )
01 70 flasher C! 39 plate ! 125
02 EMIT 0 656 C! 2 657 C!
03 ." CATCHES 0 " 1 656 C! 2 657 C!
04  ." baskets 3 " 0 656 C! 30 657
05 C! ." high " hi @ . ;
06
07 : SCREENSETUP 5 GR. 560 @ b !
08 BEGIN b @ 1 + DUP b ! C@ 66 =
09 UNTIL 70 b @ C! 6 b @ 3 + C! 6
0A b @ 4 + C! 6 b @ 5 + C!
0B STARTGAME ;
0C
0D : ORANGES DX 0 hi ! BEGIN
0E SCREENSETUP BALLSTART BEGIN
0F DELAY 764 C@ 255 = NOT IF WAIT T
10 HEN DELAY 0 STICK DUP left = IF
11 LEFTONE THEN right = IF RIGHTONE
12  THEN 0 COLOR balx @ baly @ PLOT
13  a @ 1 + DUP a ! 1 > IF baly @ 1
14  + baly ! 0 a ! THEN 1 COLOR
15 balx @ baly @ PLOT baly @ down =
16  IF CATCHCHECK THEN baskets @ 0
17 = UNTIL 200 100 DO 1 I 10 10
18 SOUND 40 1 DO DELAY LOOP 10
19 +LOOP 1 250 12 10 SOUND 254 1 DO
```

```
SCR #  23
00 ( scr# 23  empty block 1/1 ) ;S
01
02
03
04
05
06
07
08
09
0A
0B
0C
0D
0E
0F
10
11
12
13
14
15
16
17
18
19
```

```
1A  DELAY LOOP 1 0 0 0 SOUND hi @
1B catches @ < IF catches @ hi !
1C THEN 0 UNTIL ;
1D
1E
1F ;S
```


WALDEN'S C

The normal way of testing a function's speed is to call it a given number of times and see how long it took. The problem with this approach is that some functions may take a few split seconds to execute, and others could take several minutes. In the program SPEED.C, two functions are called repeatedly for a given number of seconds and then the results are compared. This way you know the computer will only be tied up for a given length of time, and you can easily test two different ways of programming the same function, and compare the results. It operates on the more is better principal; the more repetitions you get, the faster your function is. Because the computer does a lot of housekeeping chores while it's running your program, the results will vary slightly even when calling the same function. A difference of less than 10 percent between the functions will probably not be significant.

SPEED.BAS is a similar program written in BASIC. For a lot of reasons (too many to explain in this article) the timing run is limited to 4 seconds for each function. Because BASIC is slow to begin with, you will not be able to put in complex functions and have it run in under 4 seconds. This program is mainly for comparison to C. However, I did run it on a variety of BASIC's, and you may be interested in the results. This benchmark calls a null function (all it does is return).

Atari BASIC - 267; BASIC XL - 433/714 (normal/fast mode); BASIC XL with MMG FASTCHIP - 550/876; BASIC XE - 349/603/972 (normal/extensions/fast); DEEP BLUE C - 1,237; DMC/65 - 4,807.

```
100 REM SPEED.BAS - TEST FUNCTION SPEE
D
110 GOTO 160:REM MAIN LOOP
120 REM TEST1
130 RETURN :REM PLACE FUNCTION HERE, A
ND END WITH A RETURN
140 REM TEST2
150 RETURN :REM PLACE FUNCTION HERE, A
ND END WITH A RETURN
160 REP1=0:REP2=0
170 POKE 20,0:REM CLEAR SYSTEM CLOCK
180 GOSUB 130:REP1=REP1+1:IF (PEEK(20)
(240) THEN 180:REM RUNS FOR 4 SECONDS
190 POKE 20,0:REM CLEAR SYSTEM CLOCK
200 GOSUB 130:REP2=REP2+1:IF (PEEK(20)
(240) THEN 200:REM RUNS FOR 4 SECONDS
210 ? :? "TEST1 = ";REP1;" repetitions
"
220 ? :? "TEST2 = ";REP2;" repetitions
"
230 ?
240 IF (REP1)REP2) THEN ? "TEST1 is fa
ster then TEST2."
250 IF (REP2)REP1) THEN ? "TEST2 is fa
ster then TEST1."
260 IF (REP2=REP1) THEN ? "TEST2 and T
EST2 are the same"
```

FURTRADER FROM UK ATARI USERS GROUP

```
5 OPEN #1,4,0,"K:"
6 HK=5:CF=50:OF=100:RF=200:H=7:D=7:F=5
:M=100:T=0
7 ? "{CLEAR}"
17 TRAP 10000:GOTO 7000
25 GRAPHICS 0
30 POSITION 0,3:? "YOUR COMMANDS ARE :
"
40 ? :? "(1) LOOK FOR TRADING POST."
50 ? :? "(2) GO TO BED."
60 ? :? "(3) EAT/DRINK FOOD AND WATER.
"
70 ? :? "(4) FOOD/HUNGER/REST/STORES C
K."
90 POSITION 0,14:? "COMMANDS AT TRADIN
G POST ONLY :"
130 ? :? "(5) SEE GOING EXCHANGE RATE.
"
140 ? :? "(6) MAKE A DEAL."
150 POKE 764,255:POSITION 0,21:? "WHAT
 IS YOUR COMMAND ?":INPUT A
170 IF INT(A)<>A OR A<1 OR A>6 THEN 90
20
180 IF T<>1 AND A>4 THEN 9040
185 IF A=6 AND ERV=0 THEN 9000
190 ? "{CLEAR}"
200 GOTO (A*1000)
1000 IF INT((RND(1)*10)+1)>6 THEN 1500

1010 T=1
1020 POSITION 0,8:? "CONGRATULATIONS .
 YOU HAVE FOUND A             TRADING P
OST.":?
1025 A=INT(RND(1)*2)+1
1035 IF A=1 THEN F=F+1:? "YOU ALSO MAN
AGED TO REFILL YOUR STORE          OF F
OOD."
1045 GET #1,Y:GOTO 17
1500 T=0:POSITION 0,11:? "BAD LUCK.YOU
 DID NOT MANAGE TO FIND A     TRADING
POST ANYWHERE ."
1515 GET #1,Y:GOTO 17
2000 A=INT(RND(1)*5)+1
2005 IF A<>1 THEN D=7
2010 POSITION 0,10:? "NIGHT,NIGHT.DON'
T LET THE BUGS BITE."
2011 GET #1,Y
2012 IF A=1 THEN POSITION 0,15:? "THE
BED BUGS BIT AND YOU DID NOT MANAGE TO
 GET ANY SLEEP !!"
2013 IF A=1 THEN POSITION 0,17:? "":PO
SITION 18,5:? "  ":FO
R AA=1 TO 1000:NEXT AA
2020 GOTO 17
3000 IF F<>0 THEN POSITION 0,11:? "YOU
 HAVE NOW TAKEN THE LOAD OFF ONE
     PACKHORSE.":H=7
3020 IF F=0 THEN POSITION 0,11:? "BAD
LUCK,THERE IS NO FOOD LEFT."
3030 IF F<>0 THEN F=F-1:GET #1,Y
3040 GOTO 17
4000 ? "FOOD / HUNGER / REST / STORES
CK"
4010 POSITION 0,3:? "YOU HAVE ";HF;" H
AGGIS FURS."
4030 ? :? "YOU HAVE ";CF;" COYOTE FURS
."
4050 ? :? "YOU HAVE ";OF;" OCELOT FURS
."
4070 ? :? "YOU HAVE ";RF;" RABBIT FURS
."
4080 POSITION 0,12:? "YOU CAN SURVIVE
";D;" COMMANDS WITHOUT ANY FOOD."
4100 ? :? "YOU HAVE ";F;" PACKHORSES O
F FOOD AND WATER."
4120 ? "YOU CAN SURVIVE ";D;" COMMANDS
 WITHOUT ANY REST."
4125 POSITION 0,21:? "YOU NOW HAVE &";
M
4130 GET #1,Y
4140 GOTO 17
5000 IF ERV=1 THEN 5040
5005 HFP=INT(RND(1)*40)+61
5010 CFP=INT(RND(1)*30)+31
5020 OFP=INT(RND(1)*20)+11
5030 RFP=INT(RND(1)*10)+1
5040 POSITION 0,1:? "EXCHANGE RATE :"
5050 POSITION 0,4:? "HAGGIS FUR = &";H
FP
5060 POSITION 0,8:? "COYOTE FUR = &";C
FP
5070 POSITION 0,12:? "OCELOT FUR = &";
OFP
5080 POSITION 0,16:? "RABBIT FUR = &";
RFP
5090 GET #1,Y
5095 ERV=1
5100 GOTO 17
6000 POSITION 0,4:? "YOU HAVE ";HF;"
  HAGGIS FURS"
6010 POSITION 0,8:? "YOU HAVE ";CF;"
  COYOTE FURS"
6020 POSITION 0,12:? "YOU HAVE ";OF;"
   OCELOT FURS"
6030 POSITION 0,16:? "YOU HAVE ";RF;"
   RABBIT FURS"
6035 POSITION 0,0:? "YOU NOW HAVE A TO
TAL OF &";M
6040 POSITION 0,20:? "HOW MANY HAGGIS
FURS WILL YOU SELL ?"
6045 IF HF<=0 THEN 6090
6050 INPUT X:X=INT(X)
6060 IF X>HFORX<0 THEN 6050
6070 M=M+(X*HFP):POSITION 25,0:? M;"
"
6080 HF=HF-X
6085 POSITION 9,4:? HF;"  "
6090 POSITION 9,20:? "COYOTE FURS"
6095 IF CF<=0 THEN 6140
6100 INPUT X:X=INT(X)
6110 IF X>CFORX<0 THEN 6100
6120 M=M+(X*CFP):POSITION 25,0:? M;"
"
6130 CF=CF-X
6135 POSITION 9,8:? CF;"  "
6140 POSITION 9,20:? "OCELOT FURS"
6145 IF OF<=0 THEN 6180
6150 INPUT X:X=INT(X)
6153 IF X>OFORX<0 THEN 6150
6160 M=M+(X*OFP):POSITION 25,0:? M;"
"
6170 OF=OF-X
6175 POSITION 9,12:? OF;"  "
6180 POSITION 9,20:? "RABBIT FURS"
6185 IF RF<=0 THEN 6220
6190 INPUT X:X=INT(X)
6195 IF X>RFORX<0 THEN 6190
6200 M=M+(X*RFP):POSITION 25,0:? M;"
"
6210 RF=RF-X
6215 POSITION 9,16:? RF;"  "
6220 POSITION 9,20:? "HAGGIS FURS WILL
 YOU BUY ?"
6225 IF (M<HFP) THEN 6270
6230 INPUT X:X=INT(X)
6240 IF (X*HFP)>MORX<0 THEN 6230
6250 M=M-(X*HFP):POSITION 25,0:? M;"
"
6260 HF=HF+X:POSITION 9,4:? HF;"  "
6270 POSITION 9,20:? "COYOTE FURS"
6275 IF (M<CFP) THEN 6320
6280 INPUT X:X=INT(X)
6290 IF (X*CFP)>MORX<0 THEN 6280
6300 M=M-(X*CFP):POSITION 25,0:? M;"
"
6310 CF=CF+X:POSITION 9,8:? CF;"  "
6320 POSITION 9,20:? "OCELOT FURS"
6325 IF (M<OFP) THEN 6370
6330 INPUT X:X=INT(X)
6340 IF (X*OFP)>MORX<0 THEN 6330
6350 M=M-(X*OFP):POSITION 25,0:? M;"
"
6360 OF=OF+X:POSITION 9,12:? OF;"  "
6370 POSITION 9,20:? "RABBIT FURS"
6375 IF (M<RFP) THEN 6420
6380 INPUT X:X=INT(X)
6390 IF (X*RFP)>MORX<0 THEN 6380
6400 M=M-(X*RFP):POSITION 25,0:? M;"
"
6410 RF=RF+X:POSITION 9,16:? RF;"  "
6415 T=0:ERV=0:GOTO 17
7000 ? "{CLEAR}":D=D-1:H=H-1
7020 IF H<0 THEN 8020
7030 IF D<0 THEN 8000
7040 IF INT(RND(1)*125)>5 THEN 7200
7050 HF=HF-INT(RND(1)*(HF/2))
7070 CF=CF-INT(RND(1)*(CF/2))
7090 OF=OF-INT(RND(1)*(OF/2))
7110 RF=RF-INT(RND(1)*(RF/2))
7130 POSITION 0,9:? "SOME ROBBERS JUST
```

ST LIBRARY

We have some additions to our ST Library: 3 new Basic programs: Calc.bas, Labels.bas and Title.bas. The Calc program is a 4-function calculator.

One new ST Writer application: STWCON.TOS and XYZZX.TXT. This program purports to be able to configure ST Writer for whichever printer you may desire to use.

Two new sound files: Newsin.prg (draws sine waves and makes sound at the same time — demo), and Evita.sng (to be used with the MIDIDEMO.PRG — you also need a synthesizer!)

One new ST Paint program: Degasneo.prg. This one will translate picture files drawn with DEGAS into a format usable by ST Paint.

One game: MEGAROIDS (Megaroid.prg and Megaroid.rsc). This is an implementation of the classic Asteroids game written in MegaMax C. It's a good advertizement for the MegaMax product.

MORE RUMORS: Atari has sold 90,000 STs. The ST is the best-selling computer in France and Germany. The Atari ST is outselling Macintosh 6:1 and C-Amiga 30:1. Is multitasking important to you? The Atari ST hardware supports multitasking as well as any 68k system. OS/9-68k is being ported over. Atari is working on a multitasking OS. DRI is shippiing to OEMs its Concurrent DOS 68k.

The Atari ST is expandable to 256 megabytes of RAM through its DMA port (SASI compatible). Is this expandable enough for you?

Les Ellingham, of Page 6 in the United Kingdom, says their version of GEMDOS "has a bug which prevents opening folders with 8-letter file names." I'm surprised. Come on, Atari, let's get that TOS on ROMs!

PC Inter/Comm (Mark of the Unicorn, $99 discounted) is a great telecommunications program. It's as good, if not better, than anything I've seen for the IBM PC. In fact, I believe it was ported over from the IBM PC version. It's entirely menu-driven from the keyboard, or you can bypass many menu selections directly with commands as you learn them.

There are probably more than 100 commands and options and parameters you may select while using this program. The best features of this program include the ability to set up configuration files for any system you may call. Just load in a pre-configured set up and press Alt-D RETURN and the phone dials away. A version of the Xmodem protocol is available which uses cyclical redundancy checking. It will also use the Kermit protocol, as well as "Raw" and "ASCII".

One really marvelous feature is demonstrated by the way in which I now transmit the ACE Newsletter file to the typesetter. I go to bed one night after having run up PC Inter/Comm and loading in my "Callup.ic" file. I type Alt-T RETURN and go to sleep. At 2.30 am, the Atari ST takes the phone off-hook (oh yes, I had to turn on the modem, too), dials the number. Any log-on procedure is completed (which may include the exchange of passwords), the file is uploaded (a file may be downloaded instead, or in addition?). When finished, the logoff protocol is executed, the phone put back on hook, and the Atari ST goes back to sleep! What do you think of that? Two Atari STs, running this same software could each perform these tasks without a human at either end! As you can tell, I'm really bowled over by this feature.

If you're on a BBS and you're reading messages or bulletins, sometimes you might say to yourself, "I wish I had my buffer open. I wanted to save that!" This program puts the history of your terminal session into a buffer. You can page backwards through this history, even if it's been scrolled off the page. You can edit and save parts of the history.

The program emulates a DEC VT102 terminal. The F1-F4 keys function just like the VT102 unless you define new values to them. You may also assign values to F5-F10 and Shift F1-Shift F-10. Some words or phrases you use alot? Just assign them to a function key. Passwords, names, city and state, etc. are my candidates.

The manual includes about 130 pages, including an extensive table of contents. The material is well-indexed. The program is a bit spendy for most Atari users, probably. But if you can spring for the money, you won't regret it. One glaring defect: You are not able to read a disk directory from within the program(!?) I can't imagine how this feature was omitted. I've not yet seen a terminal program which didn't have this ability. I'm sure they will add this feature in the next update, and in fact I've heard they intend to do just that thing. The only other thing I can think of to add to this program is to permit the user to select colors for the screen, or perhaps to play some background music while the program's running. Maybe it can be made to turn on the microwave and warm up a snack. . . .

InSoft Newsletter (1834 Beacon St., #1, Brookline, MA 02146 617-739-9012. $45/6 issues; $70/12 issues) is a newsletter on disk devoted entirely to the Atari ST. The November, 1985 issue has 19 files including executable programs, source code files and document files for each program. They have news, editorials, reviews and rumors. For less than $6 an issue you get a disk as well as all the goodies. I'm unclear, but there might be a paper newsletter included. Check this one out. Its future has possibilities.

— Jim Bumpas

STuff

Neil Harris spoke to a user group back east on October 13 with some interesting items. He says Atari has a 3.5" drive which works with the XE machines. Atari is considering the feasibility of changing over to this new drive for the 8-bit line. The AMY chip for the XEM has been licensed to a third party to complete development. Atari retains the right to use the final chip in any product they choose.

Two expansion boxes for the ST he says might appear in 1986. One is an eight slot box which plugs into the DMA port. Extra memory and other goodies can make use of this item! The other is a full 32-bit computer in a box to plug into the ST. It will only add crunching power. The graphics, I/O, etc. will still be handled by the ST. Harris finished off by saying Atari has some advanced technology in their labs right now which will knock your socks off. He says it can be ready for market in a hurry, too, pointing to how fast the XE and ST lines got to market.

Next year should be very exciting for the Atari world. Can we stand it after all the excitement of 1985?

The **C-Amiga** was at our November meeting. I invited a local Commodore dealer to demonstrate the machine because of the wide-spread interest in this machine among Atarians. I must say the graphics capability of the development system they used was very impressive. I don't know why they didn't bring a production machine. I haven't seen anything on the Atari ST yet to match the best of what they showed. I do think our Atari Waterfall is better than the waterfall they showed.

But they showed an animated scene called "Robocity" with 5 independent objects moving around on the screen at once. Another animation was the eagle flying across the screen, making eagle noises. Most of the graphics demos were in low resolution (320x200). The 2 or 3 high-res demos did have a noticeable flicker at 640x400 interlaced mode, but the detail was very impressive. They did not show two of these high-res pictures in a row. They seemed to have to re-boot after each one.

One demo was a speech synthesis of 3 or 4 words which required the entire memory of the C-Amiga to produce. The sample rate must have been very high because the sound was almost audio quality!

They didn't demonstrate any applications software. They only had graphics and a couple of games which were not finished from Electronic Arts (Firefox and Arctic Fox — similar games). Asked about business applications they said to use a C-Amiga in business is a waste of a great game machine. They called it a "super-Atari". In my opinion, the display of text characters is not as good as on my color ST system. And nothing on the market touches the Atari ST monochrome system for text display.

The C-Amiga does not come with any software for you to use. You must buy it all extra. They don't have anything equivalent to the free ST Writer, ST Paint, LOGO and BASIC which we get with our ST purchase. And the C-Amiga operating system does not have a terminal emulator to use for telecommunication. How much does a C-Amiga cost? $1295 for the 256k console and 1-meg drive; $199 for the upgrade to 512k; $495 for the RGB monitor. $1,989.00 including no software. I don't think the machine has a market niche at that price. They're selling it as a game machine, but pricing it in the business range. I'm using my Atari ST in business, and waiting for some great games to play!

ST-SIG: We have 15 ST users in our local special interest group here in Eugene. About ⅓ of the group bought the ST as their FIRST computer! If this trend is general, then Sam Tramiel is correct that this machine will revitalize the pc market. About ⅓ of us are using the machine primarily or exclusively in business.

SPECIAL OFFER: If you share an ST program which you've written and you find useful and/or amusing share it with the rest of us. If you send it on a disk with a little article describing it, we'll return you a disk with your choice of material from our ST public domain library.

Here's a little task which might set you to work: Find a way to access the Time and Date functions which the OS keeps track of from within ST Basic. The first one to come up with the answer to this problem will receive in trade for executable code on a disk a copy of one of our public domain disks.

ST 5.25" DRIVES

I own and love my 520 ST. I found an easier way to hook up a 5.25" drive to it. Open the 3.5" drive and very very gently remove the ribbon cable and connectors. Replace the ribbon cable with a 4 or 5 foot long one using the original connectors. As Dave Small showed in November Antic, find pin 6 of the incoming plug. Make a jumper to wire 12 of the robbon cable past the connection to the 3.5" drive. Set up the 5.25" drive as Dave Small indicated with a 34 pin crimp-on edge connector ($4.95 at Radio Shack). Oddly, Radio Shack no longer carries 34 wire ribbon cable.

Dave Small was not quite correct in saying you can not format a 40 track drive. I hooked up an MPI52 40 track double sided drive, cutting one trace as Gary Sewell of Dallas ACE had told me. When you format a 40 track drive, the 40th track is formatted 41 times and the ST will think it can put 720k on the disk. As long as you don't try to put more than 340k (39 tracks worth) on the disk, no problem is found writing to the disk!

— W. M. Frank

TOKEN BASIC

(by Les by Gum; reprint: P.A.C.E. World, June, 1985)

One of the mystery words which keeps cropping up in Atari talk is ?KENISATION. This is a subject which is never explained. Yet it is very portant to Basic. The reason it isn't discussed is because you don't really need to understand it. (What is this man talking about? I hear you say.)

Well, tokenize is just the way the computer stores your program. If you were to type a line such as:

10 FOR I=1 TO 6

The line is not stored exactly like that. If you could look at this line in the computer memory you would not recognize it. It is complete gobble-de-gook.

How can you look at the program in memory? Well, the actual address where it is stored on an 8-bit Atari can be found using the following:

? PEEK(136)+256*PEEK(137).

Type this and press Return. The number given is the start address of your program.

Tokenisation is a matter of substituting codes for words. . . .

The Atari takes key words such as FOR, NEXT, GOTO, etc. and puts a number in memory in place of them. When you print a LIST on screen, the Atari looks at the code numbers and puts the correct word in place on the screen.

The word FOR appears in memory as the number 8, while NEXT is number 9.

Now type NEW and Return. Type in the following program:

```
10 C=0
20 D=PEEK(136)+256*PEEK(137):T=0
30 OFFSET=PEEK(S+C+2):F=0
40 ? PEEK(S+C);'' '';:F=F+1
50 IF F=OFFSET THEN 70
60 C=C+1:GOTO 40
70 T=T+10:IF T=100 THEN END
80 POKE 764,255:? :? ''PRESS KEY''
90 IF PEEK(764)=255 THEN 90
100 ? :? :C=C+1:GOTO 30
```

If you run this program it will print the program one line at a time showing you what the line looks like in storage.

To print each successive line, press Return.

The line numbers are stored in two bytes. The first is the LOW byte of the number, the second is the HIGH byte. Number 10 is simply 10 ł 0. A large line number like 18440 is first divided y 256 to get the ȝH byte. The remainder is the LOW byte:

18440/256=72.03125, so 72 is HIGH.

72*256=18432, so 18440-18432=8. 8 is the remainder and is also the LOW byte.

Try this the other way around. If you see the number 8 in the 1st byte and 72 in the 2d byte then multiply 72*256 and add 8. The answer is 18440.

The third byte of the line gives the total number of bytes in that line. this total includes the 2 bytes of the line number. The number will be 15. Count the bytes on the screen. From byte one (10) to the last byte (22) is 15.

The fourth byte is the number of bytes in the first statement. This line only has one statement (C=0), so the total is the same (15) as the previous byte. Where a line has more than one statement this fourth byte will be less than byte 3.

The fifth byte in our example is the token for the word LET (54). If you actually put the word LET in the program this number will be 6. Confused? Worry not. It is for the sake of a Listing that there are 2 numbers meaning LET. Number 6 will make the Atari print the word LET in the list. The number 54 means this is LET but don't print it out.

Byte number 6 is the token for the variable ''C''. Everytime you enter a new variable it is given a number starting at 128. As C is the fist variable it has come across, it is tokenized as 128. The next different variable it finds will be given the number 129 and so on.

Byte 7 is the token for ''='' (equal sign).

Byte 8 tells the computer a numeric constant follows.

The next 6 bytes (9 to 14) are reserved for the value of the numeric constant. So each time you create a new variable you lose 6 bytes which saves the value you assign to it. You also use some extra memory storing the variable name.

Byte 15 is the last byte of this line and the number 22 tells Atari ''that's the lot''.

Press Return to display line 20 of the program. Line 20 has two statements in it.

Byte 3 is the total bytes in the entire line.

Byte 4 is 37 which means there are 37 bytes in the first statement. If you count 37 bytes from the start you will see the number 20 which shows the end of the first statement, this being the token for colon (:).

In the second statement in this line the first byte (byte 38) is the total count of bytes from start to end. You will see it is the same as byte 3.

After each statement terminator (token 20) the Atari expects a count of the number of bytes so far plus how many are in the next statement. If this number matches byte 3 then it knows this is the last statement on the line.

With all these constant checks is it any wonder we keep getting error messages? Still it's a good idea. It does help to make you get it right before running. As far as possible anyway.

Now that you have learned all this about tokenization, what use will you put it to? I don't know any use for it either. but it's all good fun. get in there and Poke a few random numbers into memory. See what Atari makes of that! Hah! Bet it can't make head nor tail of it.

KARATEKA

by Jordan Mechner, Broderbund software

KARATEKA is a super animated game. I am not a karate expert by any means but this game is easy to play and hard to win (that is to rescue the maiden in distress).

The objective is to rescue ''MARIKO'' the maiden To be able to do this you have to fight all kinds of warriors differently skilled in different karate levels one at the time till in the end you have to fight the master, before you can open the door to rescue her.

The one thing I personally did not find too easy was that the control with the stock atari joysticks was inconsistent at best, but not to fret there is also a keyboard option for control of your fighter which I had a lot more control with, and used throughout my endeavor with the game.

Technically I think the game is superior to most games I have seen of late. The movie-like story seems well thoughtout. The characters move very smoothly, which gives the impression of a lifelike scenario.

The backgrounds scroll with your movements, to give a realistic effect. The fighting stances seem like the real thing (I dont know karate). So if you are into animation or Karate or both I highly reccomend Karateka. It really is fun.

— ROBERT COOK

```
    PutE()
    PrintE("          Perpetual Calen
dar")
    PutE()
    PrintE("                  by")
    PutE()
    PrintE("              Dale Lutz")

    PutE()
    PrintE("          Written in ACTI
ON!")
    PutE()
    PrintE("     Currently Set For Eps
on Printer")
    PutE()
    menu(1)="1. Tell Week Day from Dat
e"
    menu(2)="2. Print Out Calendar
 "
    ch=GetChoice(2,6)
    IF ch=1 THEN
      dayflag=1
      TellDay()
    ELSE
      dayflag=0
      Calendar()
    FI
  OD
RETURN
```

QUICK PRINT

Word processors are marvelous, but they have their limitations — especially for doing simple jobs. Suppose you want to address an envelope. In most cases it is easier to use a typewriter than a word processor. With AtariWriter you need to boot the program, set the margins, type the text, move to the print menu, set some of the printing parameters, and finally, print. For some word processors, the procedure is even more complicated. You may have to create a text file, save it to disk and then set up the print parameters before printing.

QUICK PRINT is written for those small printing jobs such as addressing an envelope or writing a memo. It allows you to use your computer and printer like an electronic typewriter. The main difference is that Quick Print processes one line at a time rather than just one character.

Turn on your printer before you run Quick Print. When you successfully run Quick Print, you will be prompted for the left and right margins. Type the left margin, comma, the right margin, and RETURN. Quick Print will accept numbers between 1 and 80 as legitimate input. Error trapping should take care of any goofs or slips of the fingers. With proper input you will see a colored line on the screen with the cursor at the beginning of the line. The length of the line will equal the difference between your right and left margins. If the length of your line is more than 40 characters, the colored line will be more than one screen line (assuming a 40-column screen). As you type, the position indicator will display the cursor position along the line. If you type beyond the length of the colored line, you will go outside your margins. A warning bell will sound when you are five characters from the end of your line and you will need to press the Option key to release the margin if you wish to add a few characters beyond the length of the line. If you want to reset the margins, press Select.

Suppose you want to address an envelope. Set the left margin at about 40 and the right margin at about 75. Then you will see a colored line on the screen which is 35 characters long. Position the envelope in the printer, and type the name. You may edit the name using the screen editor before hitting Return, but when you hit Return, the name will print and a new colored line will appear on the screen for the first line of the address. And so on. By pressing Option, you may type a few characters beyond the colored line without ill effects. It is similar to hitting the margin release on a regular typewriter. But if you enter too many characters beyond the margin your printer may break your line at the wrong place.

In the BASIC program listed, a vertical blank interrupt (VBI) is used to update the cursor position, to ring the warning bell and to determine when Option and Select are pressed. Subroutine 9000 POKEs the vertical blank interrupt into memory. The assembly language subroutine developed with MAC/65 is shown separately. It is for your information only as all the necessary DATA statements are included in the BASIC listing. Since this subroutine is longer than can fit into page six, I put it in page 80 which is reserved for the right-hand cartridge in the 800 and is unused in the XL series. I do not know about the XE series.

A display list interrupt in subroutine 6000 changes the colors at screen line 4. Subroutine 3000 is used to setup the margins. Subroutine 4000 opens the keyboard for input on channel #1 (with screen output) and opens the printer for output on channel #2. Subroutine 5000 redefines the character set to create the colored line. Lines 5010-5070 include a machine language subroutine to relocate the normal character set in RAM where it may then be modified. Lines 5080-5100 redefine the graphic "heart" to be the character for the colored line. The color was created in Graphics 0 by using color artifacting — that is, every other pixel was turned on. See you local popular books and magazines for articles on interrupts, character set redefinition and color artifacting.

The main loop of the program is lines 210-260. In line 210 the cursor position is defined so the cursor will be returned to the beginning of the colored line after the colored line is printed. The print statment in line 220 is needed to reposition the cursor so you can see where you are. Line 212 is needed to insure the cursor is positioned on the colored line when the text extends to the bottom of the screen.

The trickiest part of this program is synchronizing the timing of the vertical blank interrupt and the BASIC program. Line 225 is needed to insure that the bell does not ring at the wrong time and the "keyboard disable" used as the margin stop does not engage at the wrong time. Line 240 is needed to empty the buffer when Select is pressed so the printer does not print unwanted text.

Be careful if you type control characters because they may lock up your printer. If this occurs, just turn off the printer and turn it on again. As an added feature, you may use the control characters to enhance your text. For example, on my Axiom AT-100, Control-N will turn on the expanded text mode and Control-O wil turn it off. Also be careful not to move the cursor off the line you are typing or funny things might happen. Quick Print does not recognize special characters such as Tab unless your printer can interpret them properly.

I hope you find Quick Print as useful as I have.

— Gerry Wick

BRIMSTONE and ESSEX

(for the 800/130 XE)

Synapse describes these games as "electronic novels". They are very extensive text adventure games requiring 2 disk drives as well as 48K machines. The player/reader is required to read a book describing the background and introducing the characters in a fantasy plot. From there, you boot the disks following the prompts and begin. The all text play is similar to Infocom games with one big exception: with 2 disks crammed with huge data files to access, there are many more options. For example, playing ESSEX one evening I experimented at one point making 8 different responses at the same point in the game and found 8 different complete turns of the plot unfoldment. In a sense the player/reader creates the plot by what he or she chooses to input.

BRIMSTONE is a knights of the Round Table fantasy with castles, legends, damsels in distress etc. I find it relatively hard to get started, but once into it, the game moves along at an interesting clip with several surprises. It is far from predictable.

ESSEX is a space adventure/fantasy with many climactic moments. I find it harder to play than BRIMSTONE, but this could be because I make all the wrong decisions. The story is not, in my view on par with the best of INFOCOM games although it is keeping my interest.

My main criticism of these games and the concept is the relatively slow disk access. I frequently tire of the long waits. Perhaps I have been spoiled by the speed of the ST.

— Graham Smith

THE "ST" CONNECTION

Programs for the ST are gradually becoming more plentiful. A few of the new programs and the final edition of ST BASIC are worth a few remarks. However, this is by no means an in-depth review.

Hex, a strategy game from Mark of the Unicorn, is a very fine game — once you get into it far enough to understand the sometimes subtle play. It is sort of a cross between CHESS and ARCHON with a little of Q*BERT thrown in. It is hard to describe since there is really nothing else like it. The object is for you to turn the board surface green by jumping on the hexes making up the surface. This is complicated by your computer opponent who meanwhile is trying to turn the surface purp Altogether, there are 12 different opponents of varying skills and magi powers trying to undo you in 125 different levels. It is all very cleverly done and has great staying power and wonderful graphics.

Ultima II on the ST is wonderful. The GEM interface makes play very simple and straight forward. Best of all, the frequent disk I/O is lightning fast. The GEM is perfect for this kind of application. I love it!!!

The new ST BASIC is very powerful, and, if you can get used to hand-picking your way through all of the windows and the relatively slow speed, this adaptation of the language works. However, In playing with it one weekend, I am convinced that it is not for me. Atari's implimentation of the GEM Desktop in BASIC ST gets in my way. Supposedly Philon's Basic M is now being shipped so it will be interesting to compare the two basics although rumor has it that Basic M does not have graphic capabilities.

I have also been impressed with SHICED: a shape and icon Editor designed specifically for the ST and made by Monarch Development of Salem. This product is a very high quality and thorough product. With it you can design your own Icons for the desktop, design multi-color shapes, and even do some animation. Best of all, you can save your work in a variety of different source codes including "C" and Assembler. With the number of examples included on the disk you can learn how to interface your files in your programs.

— Graham Smith

SWAP MEET AMAZON PARK

Atari Computer Enthusiasts

A.C.E. is an independent, non-profit and tax exempt computer club and user's group with no connection to Atari Corp. We are a group interested in educating our members in the use of the Atari Computer and in giving the latest News, Reviews and Rumors.

All our articles, reviews and programs come from you, our members.

Our membership is world-wide; membership fees include the A.C.E. Newsletter. Dues are $14 a year for U.S., and $24 a year Overseas Airmail and include about 10 issues a year of the ACE Newsletter.

Subscription Dep't: 3662 Vine Maple Dr., Eugene, OR 97405.

**President— Dick Barkley, 2907 Wingate, Eugene, OR 97405 503-344-5843

Vice Pres— Larry Gold, 1927 McLean Blvd., Eugene, Or 97405 503-686-1490

8-bit Librarian— Chuck & Jody Ross, 2222 Ironwood, Eugene 97401 (503) 342-4133

ST Librarian— Jim Bumpas, 4405 Dillard Rd., Eugene OR 97405 503-484-4746

Editors— Mike Dunn, 3662 Vine Maple Dr., Eugene, OR 97405 503-344-6193

Jim Bumpas, 4405 Dillard Rd., Eugene, OR 97405 503-484-4746

E.R.A.C.E. (Education SIG Editor)— Nora Young, 105 Hansen Lane, Eugene, OR 97404 / 503-688-1458

Send 50c stamps or coin ($1 overseas) to the Ness' for the new, updated ACE Library List—new in May 85!

Bulletin Board
(503) 343-4352

On line 24 hours a day, except for servicing and updating. Consists of an 800 XL, 2 double-density double sided disk drives and 2 double-sided, double-density, 80-track disk drives, an Epson MX80 printer, a 1200 baud Prentice P212ST modem, running the Mindlink Bulletin Board software distributed by SofMark.

Best of ACE books

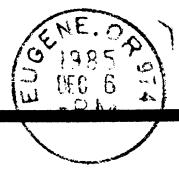
Volume 1 contains bound issues of the ACE Newsletter from the first issue, Oct 81 to June of 1982

Volume 2 covers July 1982 to June 1983

Only $12 each ($2 extra for Airmail). Available only from:
George Suetsugu
45-602 Apuapu St
Kanoehe, HI 96744

SortFinder Index

A composite index of Atari related articles in four popular computer periodicals, including ACE. Volume 1covers April, 1981 to June, 1983. Volume 2 covers July, 1983 to December, 1985. Only $6 per printed copy or $11 per disk copy for ACE members:
Jim Carr 2660 S.W. DeArmond
Corvallis, OR 97333



ATARI COMPUTER ENTHUSIASTS
3662 Vine Maple Dr Eugene OR 97405

FIRST CLASS MAIL